Journal of Digital Business and Technology Innovation (DBESTI)

https://journal.nurulfikri.ac.id/index.php/DBESTI
P-ISSN: 3047-1028 E-ISSN: 3032-775X

PEMBANGUNAN REST API UNTUK PENGELOLAAN RUTE ROBOT AGV DI PT XYZ

Fazri Egi Ramadhan[1], Imam Haromain[2], Lukman Rosyidi[3]

[1,2,3]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Depok, Jawa Barat, Indonesia 16451
fazriegi80@gmail.com, haromain@nurulfikri.ac.id, lukman@nurulfikri.ac.id

Abstract

The transportation process for automotive components at PT XYZ is still carried out manually, involving human labor, which has led PT XYZ to consider using Automated Guided Vehicle (AGV) technology to improve the efficiency and effectiveness of transportation processes. Although the implementation of AGV technology at PT XYZ can provide benefits in terms of increased efficiency and productivity, there are limitations in the AGV system. The route change process must be done manually by reprogramming each AGV one by one, which is time-consuming, labor-intensive, and increases the risk of errors and inconsistencies. This study aims to address these limitations by designing and developing a REST API to support the functionality of the AGV route management system. The method used is black box testing to evaluate the system's functionality without examining the internal code structure. The test results show that the system successfully meets user needs with a success rate of 100% from 101 test scenarios. The developed REST API can integrate AGV route data and publish and update routes through an MQTT broker. This research is expected to provide an appropriate solution to help PT XYZ realize its full potential in utilizing AGV technology.

Keywords: *Automated Guided Vehicle, Black Box Testing, MQTT, Node.js, REST API*

Abstrak

Proses transportasi komponen otomotif di PT XYZ masih dilakukan secara manual dengan melibatkan tenaga manusia, sehingga mendorong PT XYZ untuk mempertimbangkan penggunaan teknologi *Automated Guided Vehicle* (AGV) guna meningkatkan efisiensi dan efektivitas proses transportasi. Meskipun penerapan teknologi AGV di PT XYZ dapat memberikan manfaat dalam meningkatkan efisiensi dan produktivitas, terdapat keterbatasan pada sistem AGV. Proses perubahan rute harus dilakukan secara manual dengan memprogram ulang setiap AGV satu per satu, yang memakan waktu dan tenaga besar serta meningkatkan risiko kesalahan dan inkonsistensi. Penelitian ini bertujuan untuk mengatasi keterbatasan tersebut dengan merancang dan mengembangkan REST API untuk mendukung fungsionalitas sistem manajemen rute AGV. Metode pengujian yang digunakan adalah *black box testing* untuk mengevaluasi fungsionalitas sistem tanpa memeriksa struktur internal kodenya. Hasil pengujian menunjukkan bahwa sistem berhasil memenuhi kebutuhan pengguna dengan tingkat keberhasilan 100% dari 101 skenario pengujian. REST API yang dikembangkan mampu mengintegrasikan data rute AGV dan mempublikasikan serta memperbarui rute melalui MQTT *broker*. Penelitian ini diharapkan dapat memberikan solusi yang tepat untuk membantu PT XYZ mewujudkan potensi maksimalnya dalam penggunaan teknologi AGV.

Kata kunci: *Automated Guided Vehicle, Black Box Testing,* MQTT, Node.js, REST API

1. PENDAHULUAN

Pada era Revolusi Industri 4.0 telah terjadi disrupsi yang dapat diartikan sebagai perpindahan aktivitas manusia dari dunia nyata ke dunia maya melalui otomatisasi dan konektivitas gerakan non-linear dengan menggunakan robot dan teknologi [1]. Industri 4.0 yang menggunakan sistem kecerdasan buatan merupakan salah satu inovasi teknologi yang paling berpengaruh dalam meningkatkan efisiensi dan produktivitas di berbagai sektor. Salah satu inovasi teknologi tersebut adalah robot AGV (*Automated Guided Vehicle*). AGV adalah kendaraan otomatis yang dapat bergerak dan melakukan tugas tanpa memerlukan operator. AGV memiliki banyak keunggulan dibandingkan sistem penanganan material lainnya, seperti mobilitas yang fleksibel, keandalan, biaya pengoperasian yang rendah, dan kemudahan integrasi dengan sistem lain [2].

Diterima 8 Agustus 2024, Direvisi 28 Oktober 2024, Diterima untuk publikasi 29 Oktober 2024

PT XYZ merupakan perusahaan manufaktur terkemuka di Indonesia dalam industri komponen otomotif. PT XYZ menjalankan model bisnis tiga pilar yang terdiri dari Konversi Kendaraan, Suku Cadang, dan Bisnis Cetakan. Proses transportasi komponen otomotif di PT XYZ masih dilakukan secara manual dengan melibatkan tenaga manusia. Hal ini mendorong PT XYZ untuk mempertimbangkan penggunaan teknologi AGV untuk meningkatkan efisiensi dan efektivitas proses transportasi.

Berdasarkan hasil wawancara, meskipun penerapan teknologi AGV di PT XYZ dapat memberikan manfaat dalam meningkatkan efisiensi dan produktivitas, namun terdapat keterbatasan dalam sistem AGV, yaitu proses perubahan rute harus dilakukan secara manual dengan memprogram ulang setiap AGV satu per satu. Proses ini memakan waktu dan tenaga yang besar, serta meningkatkan risiko kesalahan dan inkonsistensi.

Untuk mengatasi keterbatasan tersebut, penulis melakukan penelitian dengan judul "Rancang Bangun REST API untuk Manajemen Rute Robot AGV di PT XYZ" yang bertujuan untuk merancang dan membangun REST API untuk manajemen rute robot AGV di PT XYZ, serta mengintegrasikan REST API tersebut dengan sistem AGV. Ruang lingkup dan fokus dalam penelitian ini adalah perancangan REST API untuk mendukung sistem manajemen rute robot AGV. Objek dalam penelitian ini adalah PT XYZ, dan pengembangan REST API dilakukan berdasarkan kebutuhan spesifik perusahaan. Pengujian fungsionalitas REST API dilakukan menggunakan *Postman* dan MQTT Explorer sebagai alat bantu. Adapun rumusan masalah dalam penelitian ini adalah bagaimana merancang dan membangun REST API untuk manajemen rute robot AGV, serta bagaimana cara mengintegrasikan REST API dengan sistem AGV. Penelitian ini terbatas pada area operasional PT XYZ dan difokuskan pada pengembangan REST API untuk mendukung sistem pengelolaan rute robot AGV di perusahaan tersebut.

AGV (*Automated Guided Vehicle*)

AGV (*Automated Guided Vehicle)* adalah kendaraan yang beroperasi secara otomatis dengan menggunakan sistem navigasi untuk menuju tujuan yang telah ditentukan. AGV digunakan secara luas di berbagai industri manufaktur untuk melakukan pemindahan produk. Fungsinya mirip dengan *lift-truck* yang dikemudikan oleh manusia, namun AGV beroperasi secara otomatis tanpa intervensi manusia [3].

REST API

Roy Fielding dari *University of California* menciptakan istilah *Representational State Transfer* (REST). Layanan web ini sangat ringan dan mudah. Ide utama dari desain REST adalah kinerja, skalabilitas, kesederhanaan, portabilitas, dan kemampuan untuk dimodifikasi. REST adalah arsitektur layanan web klien server, di mana klien melakukan permintaan kepada server, server memproses permintaan, dan kemudian mengirimkan respons. Aplikasi web yang memanfaatkan arsitektur REST disebut sebagai layanan web RESTful. Metode HTTP GET, POST, PUT, dan DELETE digunakan oleh layanan web RESTful untuk menghasilkan, menerima, membuat, memperbarui dan menghapus sumber daya [4]. API adalah sebuah antarmuka yang memungkinkan pengguna untuk menghubungkan dan mengintegrasikan data serta aplikasi yang beroperasi pada platform yang berbeda [5].

Node.js

Node.js adalah *runtime environment* JavaScript yang bersifat *cross-platform* dan *open-source*. Node.js memungkinkan kode JavaScript dijalankan di mana saja, tidak hanya di browser. Node.js memiliki kinerja yang luar biasa karena dibangun dengan V8 JavaScript Engine milik Google. Arsitektur *Single Threaded Event Loop* memungkinkan Node.js untuk menjalankan banyak proses sekaligus. Model pemrosesan Node.js bergantung pada model *event based* dan mekanisme *callback* JavaScript [6].

PostgreSQL

Sebuah sistem basis data *relational* seperti PostgreSQL, dirancang untuk membantu pengguna dalam mengelola dan memahami hubungan antar data. Sebagai salah satu basis data *relational open-source* terkemuka, PostgreSQL telah mengalami pengembangan selama lebih dari 30 tahun, menjadikannya salah satu pilihan yang paling mapan dalam dunia basis data *relational*. Keunggulan PostgreSQL terletak pada fleksibilitas dan integritasnya yang luar biasa, yang membuatnya populer di kalangan pengembang dan administrator. Misalnya, PostgreSQL mendukung kueri *relational* dan *non-relational*, sementara sifat sumber terbukanya memungkinkan komunitas pengembang untuk terus meningkatkan sistem basis data ini [7].

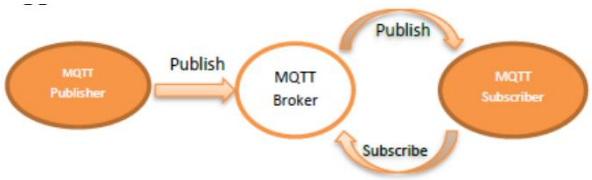
Postman

Postman adalah sebuah aplikasi yang digunakan untuk menguji API. Aplikasi ini memfasilitasi pengiriman permintaan dari klien ke server web dan menerima respons dalam berbagai format. Penggunaannya tidak memerlukan kerangka kerja tambahan atau pengaturan khusus, sehingga proses pengujian aplikasi dengan *Postman* menjadi lebih mudah dan efisien. *Postman* memiliki antarmuka yang sederhana dan mudah digunakan, serta menawarkan berbagai fitur yang mempermudah proses kerja [8].

MQTT (*Message Queuing Telemetry Transport)*

MQTT (*Message Queuing Telemetry Transport*) merupakan jenis protokol jaringan yang digunakan di IoT (*Internet of Things*) yang berfungsi sebagai komunikasi yang bersifat *machine to machine* atau M2M. Dapat beroperasi pada *bandwidth* yang sangat cocok untuk digunakan pada aplikasi IoT yang membutuhkan energi dan media transmisi data minimal, serta ringan dalam transmisi

pesan ringan dalam header dengan ukuran kecil yaitu 2 bytes [9].



Gambar 1. Cara kerja MQTT [9]

Pada protokol MQTT seperti yang bisa dilihat di gambar 1, *publish* berfungsi sebagai pengirim pesan, dan *subscribe* berfungsi sebagai penerima pesan. Dalam protokol MQTT, terdapat broker yang berfungsi sebagai *topic* yang dikirmkan oleh *publisher* untuk di teruskan ke *subscriber* berdasarkan permintaan dari pengguna [9].

MQTT Explorer

MQTT Explorer adalah klien MQTT komprehensif yang memberikan ikhtisar terorganisir tentang topik MQTT dan menyederhanakan penggunaan alat/layanan *broker* [10].

UML (*Unified Modeling Language*)

UML adalah sebuah bahasa yang digunakan untuk menggambarkan, memvisualisasikan, merancang, dan mendokumentasikan sistem informasi. Awalnya, UML digunakan untuk pemahaman dan dokumentasi sistem informasi, tetapi sekarang penggunaannya telah meluas di berbagai sektor industri. UML dikenal sebagai standar terbuka yang menjadi bahasa pemodelan umum dalam pengembangan perangkat lunak dan sistem [11].

Pengujian *Black Box*

Pengujian *Black Box* adalah metode pengujian yang berfokus pada spesifikasi fungsional perangkat lunak tanpa memperhatikan implementasi internalnya. Dalam pengujian ini, penguji menetapkan serangkaian kondisi input dan menguji program berdasarkan spesifikasi fungsionalnya. Tujuan utama dari *Black Box Testing* adalah untuk menguji fungsi-fungsi perangkat lunak dan memastikan bahwa data masukan diolah dengan benar sesuai dengan harapan, serta memastikan bahwa informasi yang disimpan dan dipertahankan secara eksternal selalu *up-to-date*. Jenis kesalahan yang biasa diidentifikasi meliputi kesalahan fungsi, kesalahan antarmuka, kesalahan struktur data dan basis data, kesalahan performa, serta kesalahan inisialisasi dan terminasi [12].

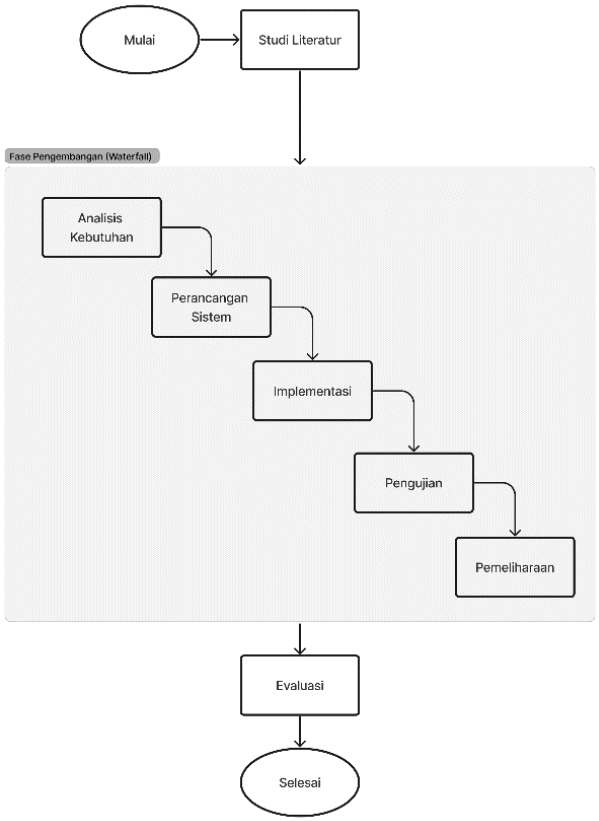
Metodologi *Waterfall*

Metode *waterfall* merupakan sebuah model pengembangan perangkat lunak yang mengikuti serangkaian tahapan secara berurutan dari awal hingga akhir. Model ini sebenarnya dikenal sebagai "*Linear Sequential Model*" yang menggambarkan pendekatan yang sistematis dan terstruktur terhadap pengembangan sistem yang berskala besar. Proses dimulai dengan spesifikasi pengguna dan berlanjut melalui tahapan perencanaan, pemodelan, konstruksi, dan implementasi sistem, sebelum akhirnya mencapai tahap pengujian sistem [13].

2. METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini dilakukan dengan menggunakan beberapa tahapan seperti pada gambar 2 berikut.



Gambar 2. Tahapan Penelitian

1. Studi Literatur

Pada studi literatur, penulis melakukan pembelajaran mendalam terkait dengan penelitian dan merujuk kepada berbagai referensi seperti artikel, jurnal, dan penelitian lainnya. Melalui proses ini, penulis dapat memperdalam pemahaman terhadap metode-metode yang relevan untuk mengatasi permasalahan yang menjadi fokus penelitian.

2. Analisis Kebutuhan

Pada tahap ini, dilakukan analisis mendalam terhadap kebutuhan pengguna dan tujuan sistem yang akan dikembangkan. Tujuannya adalah untuk memperoleh pemahaman yang menyeluruh mengenai kebutuhan fungsional dan non-fungsional.

3. Perancangan Sistem

Tahap ini bertujuan untuk menghasilkan solusi yang sesuai dengan kebutuhan yang telah diidentifikasi pada tahap analisis kebutuhan. Perancangan mencakup pembuatan

arsitektur sistem, desain *database*, serta spesifikasi teknis lainnya yang akan menjadi dasar untuk implementasi sistem.

4. Implementasi

Pada tahap ini, penulis mengimplementasikan desain sistem menjadi kode program yang dapat dijalankan. Kode program yang dikembangkan sesuai dengan spesifikasi yang telah ditetapkan dalam tahap sebelumnya.

5. Pengujian

Tahap ini bertujuan untuk memverifikasi bahwa sistem yang dibangun beroperasi sesuai dengan kebutuhan dan spesifikasi yang telah ditetapkan sebelumnya. Pengujian dilakukan untuk mengidentifikasi serta memperbaiki bug atau kesalahan dalam sistem.

6. Pemeliharaan

Pada tahap ini, dilakukan pemeliharaan sistem yang meliputi instalasi dan proses perbaikan sistem jika terdapat kesalahan atau *bug* yang tidak terdeteksi selama pengujian.

7. Evaluasi

Pada tahap ini, penulis menarik kesimpulan dari hasil penelitian yang telah dilakukan serta memberikan saran atau rekomendasi untuk pengembangan lebih lanjut.

2.2 Rancangan Penelitian

Pada tahap ini akan dilakukan penyusunan sebagai tahap awal dalam merinci langkah-langkah yang dilakukan pada penelitian ini, yaitu dengan wawancara yang dilakukan dengan cara penyampaian sejumlah pertanyaan peneliti terhadap pengembang IoT dan manajemen dari vendor yang bekerja sama dengan PT XYZ untuk mendapatkan informasi yang akurat. Kemudian, studi dokumen dilakukan untuk mengumpulkan data yang sudah ada dalam catatan dokumen, dan berfungsi sebagai pendukung dan pelengkap bagi data yang diperoleh melalui wawancara. Dari dua sumber tersebut, data diperoleh untuk dijadikan *requirement* yang disusun berdasarkan tingkat prioritas dari *requirement* tersebut. Jenis penelitian yang digunakan adalah penelitian pengembangan, atau *Research and Development* (R&D). Pengembangan dilakukan dengan merancang dan membangun REST API berdasarkan kebutuhan pengguna. Kemudian metode pengumpulan data dilakukan dengan wawancara serta studi dokumen. Lingkungan pengembangan dilakukan di rumah peneliti serta *tools* yang digunakan yaitu laptop MSI GF63 *Thin*, *Visual Studio Code*, Node.js, PostgreSQL, *Postman*, MQTT *Explorer*, *Figma*, dan *draw.io*.

3. HASIL DAN PEMBAHASAN

Pada bagian ini akan dibahas mengenai analisis dan proses perancangan serta pengembangan REST API untuk manajemen rute robot AGV di PT XYZ.

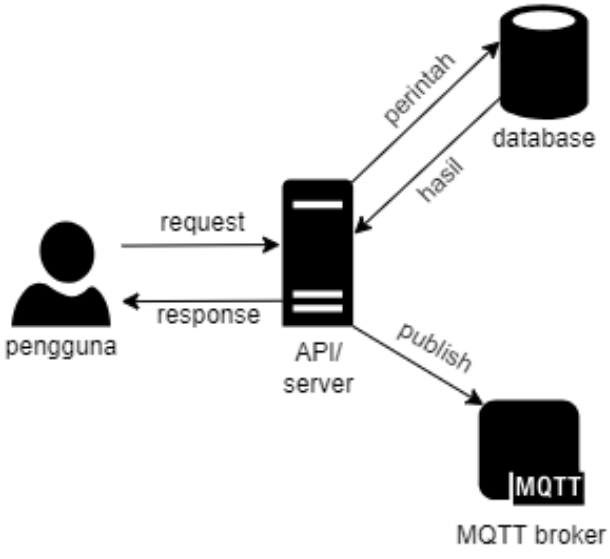
3.1 Analisis Kebutuhan

Analisis kebutuhan diperlukan untuk menetapkan kebutuhan dalam pengembangan REST API untuk pengelolaan rute AGV. Data diperoleh melalui wawancara yang dilakukan dengan pengembang IoT dan manajemen dari vendor yang bekerja sama dengan PT XYZ, serta studi dokumen untuk mengumpulkan informasi yang sudah ada dalam catatan dokumen. Berikut adalah hasil analisisnya:

1. Kebutuhan Fungsional
   - Sistem harus memungkinkan pengguna untuk *register* dan *login*.
   - Sistem harus memungkinkan pengguna untuk mengelola data pengguna.
   - Sistem harus memungkinkan pengguna untuk mengelola data AGV.
   - Sistem harus memungkinkan pengguna untuk mengelola kode pergerakan AGV.
   - Sistem harus memungkinkan pengguna untuk mengelola data RFID.
   - Sistem harus memungkinkan pengguna untuk mengelola rute AGV.
2. Kebutuhan Non-Fungsional
   - Memastikan hanya pengguna yang diotorisasi yang dapat mengakses sistem.
   - Sistem harus merespons perintah dan perubahan kondisi dengan cepat.

3.2 Perancangan Sistem

3.2.1 Rancangan Arsitektur

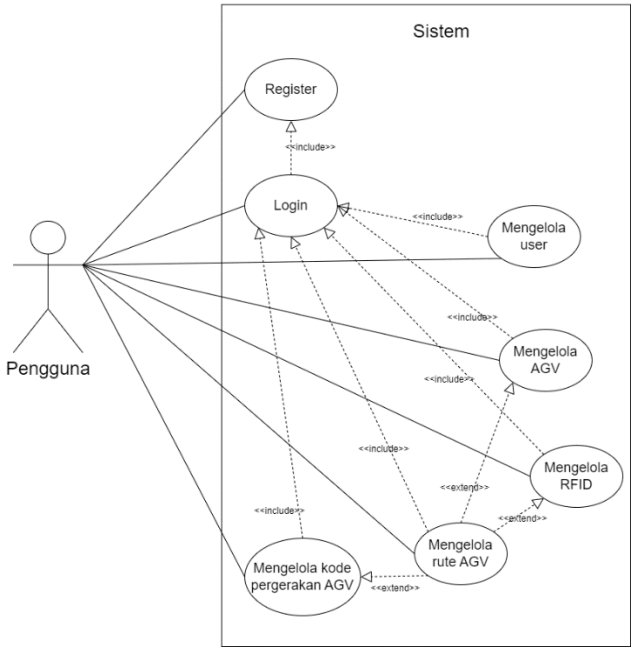


Gambar 3. Rancangan Arsitektur

Dalam REST API pengelolaan rute AGV seperti yang dilihat pada gambar 3, alur komunikasi antara pengguna, API/server, *database*, dan *broker* MQTT tergambar dalam rancangan arsitektur di atas. Pengguna mengirimkan permintaan (*request*) ke API/server, yang kemudian memproses permintaan tersebut dan memberikan tanggapan kembali (*response*) kepada pengguna. API/server juga

berkomunikasi dengan *database* untuk menyimpan atau mengambil data yang diperlukan melalui perintah tertentu dan menerima hasilnya. Selain itu, API/server berinteraksi dengan *broker* MQTT untuk mempublikasikan (*publish*) pesan yang digunakan oleh sistem AGV. Ini memastikan bahwa informasi yang dibutuhkan oleh AGV untuk navigasi dan operasi lainnya selalu tersedia dan terkini.

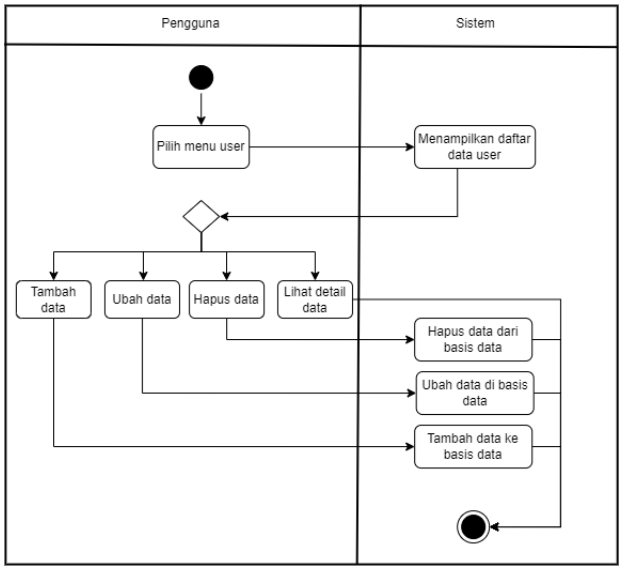
3.2.2 *Use Case Diagram*



Gambar 4. *Use Case Diagram*

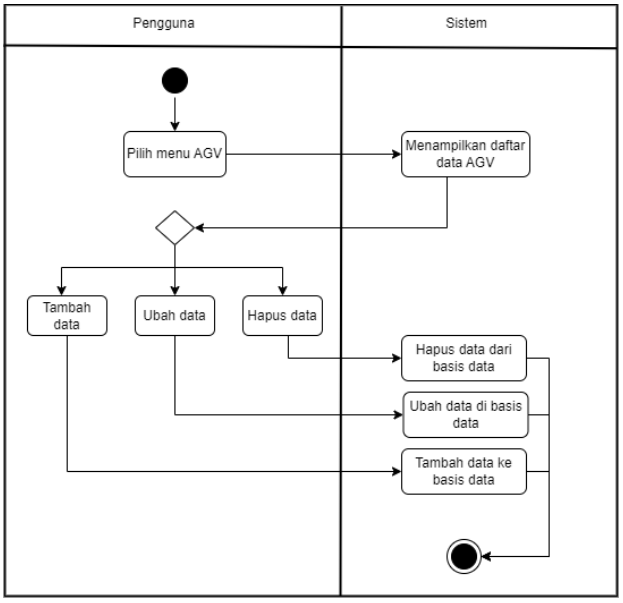
Gambar 4 *use case diagram* di atas menunjukkan interaksi antara pengguna dan sistem pengelolaan rute AGV. Pengguna dapat melakukan berbagai hal di dalam sistem, seperti mendaftarkan akun baru dan masuk ke dalam sistem. Setelah masuk, pengguna memiliki kemampuan untuk mengelola pengguna termasuk melihat profil diri sendiri, menambahkan pengguna baru, mengubah profil diri sendiri atau pengguna lain, dan menghapus pengguna lain. Pengguna juga memiliki kemampuan untuk mengelola AGV, termasuk menambah, melihat, mengubah, dan menghapus data unit AGV dari sistem. Selain itu, pengguna dapat mengelola kode pergerakan AGV, yang mencakup definisi kode pergerakan yang digunakan AGV untuk navigasi. Pengguna juga dapat mengelola RFID, yang memungkinkan pengguna untuk menambah, melihat, atau menghapus informasi RFID yang digunakan AGV untuk navigasi dan identifikasi. Terakhir, pengguna dapat mengatur rute AGV untuk memastikan setiap AGV mengikuti jalur yang telah ditentukan sesuai dengan kebutuhan operasional. Diagram ini menunjukkan berbagai fungsi penting yang harus dapat diakses dan dikendalikan oleh pengguna untuk memastikan operasi AGV berjalan dengan efisien dan efektif.

3.2.3 *Activity Diagram*



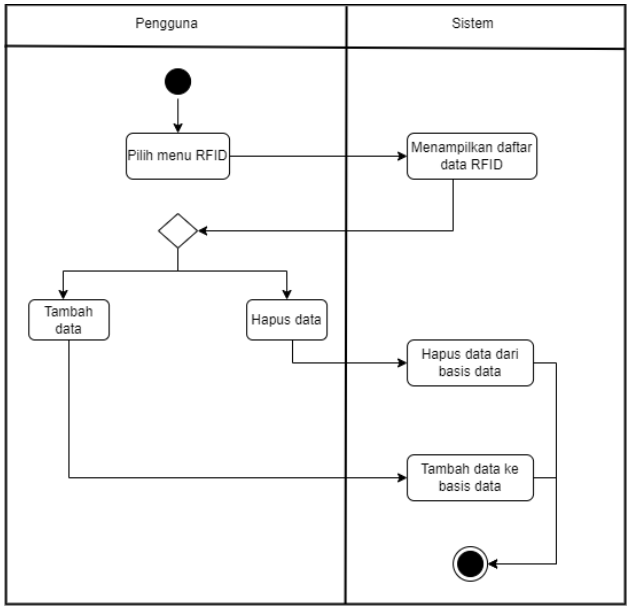
Gambar 5. *Activity Diagram* Kelola Pengguna

Pada *activity diagram* kelola akun pengguna, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 5 di atas.



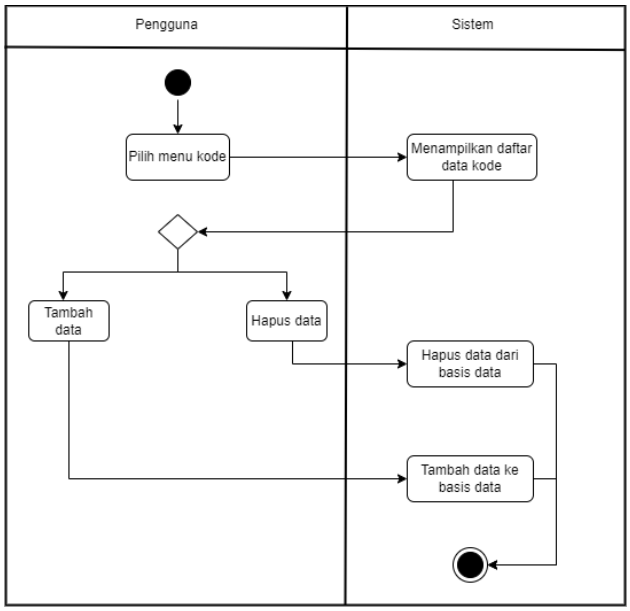
Gambar 6. *Activity Diagram* Kelola AGV

Pada *activity diagram* kelola AGV, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 6 di atas.



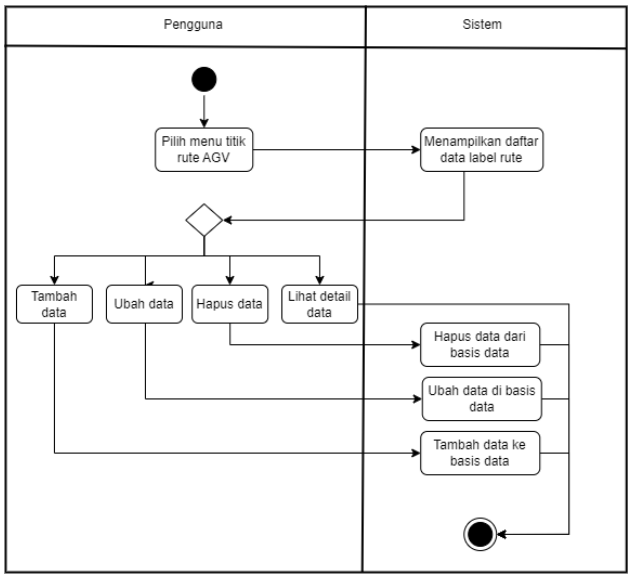
Gambar 7. *Activity Diagram* Kelola RFID

Pada *activity diagram* kelola RFID, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 7 di atas.



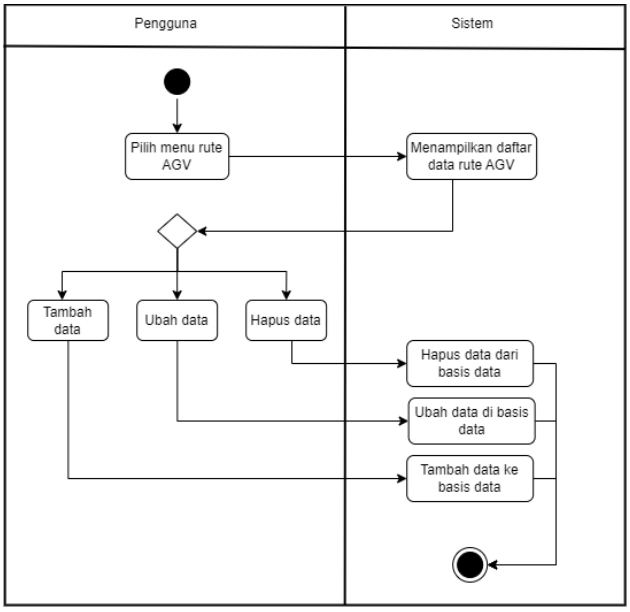
Gambar 8. *Activity Diagram* Kelola Kode Pergerakan

Pada *activity diagram* kelola kode pergerakan, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 8 di atas.



Gambar 9. *Activity Diagram* Kelola Rute

Pada *activity diagram* kelola rute, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 9 di atas.



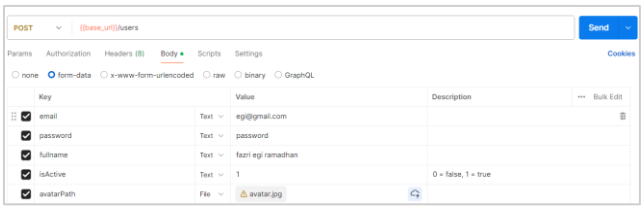
Gambar 10. *Activity Diagram* Kelola Rute AGV

Pada *activity diagram* kelola rute AGV, terdapat alur proses antara pengguna dengan sistem yang dapat dilihat pada Gambar 10 di atas.

3.3 Implementasi

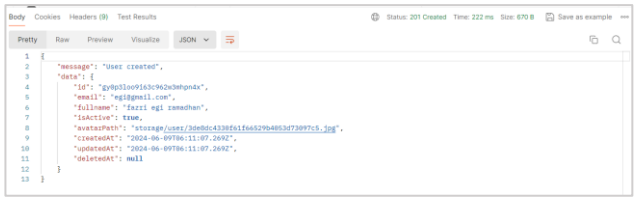
Implementasi yang dilakukan penelitian ini menghasilkan REST API yang mencakup beberapa fitur untuk manajemen rute robot AGV.

1. Menambahkan Akun Pengguna (*Register*)



Gambar 11. Permintaan *Register*

Pada gambar 11 di atas, terlihat contoh permintaan (*request*) untuk mendaftarkan pengguna menggunakan metode POST ke *endpoint* `{{base_url}}/users`. Pada permintaan ini membutuhkan *request body* yang berisi beberapa data yang diperlukan. Permintaan ini mengirimkan data untuk membuat pengguna baru. Server akan memproses data dan memberikan respons berdasarkan hasilnya. Jika permintaan berhasil, server akan menghasilkan respons yang menunjukkan bahwa akun pengguna telah berhasil dibuat.



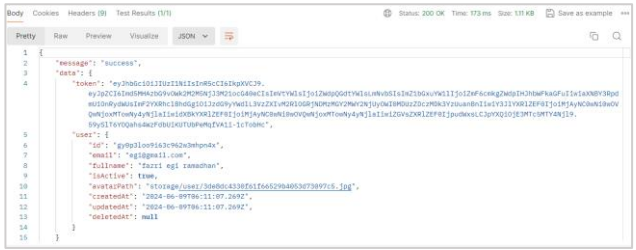
Gambar 12. Respons Berhasil *Register*

Gambar 12 di atas merupakan respons server jika permintaan *register* berhasil.

2. *Login*

Gambar 13. Permintaan *Login*

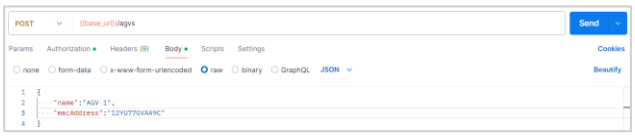
Pada gambar 13 di atas, terlihat contoh permintaan (*request*) untuk *login* menggunakan metode POST ke *endpoint* `{{base_url}}/web-admin/auth/login`. Pada permintaan ini membutuhkan *request body* yang berisi beberapa data yang diperlukan. Permintaan ini mengirimkan data untuk *login*. Server akan memvalidasi data dan memberikan respons berdasarkan hasilnya. Jika permintaan berhasil, server akan menghasilkan respons yang menunjukkan bahwa *login* telah berhasil.



Gambar 14. Respons Berhasil *Login*

Gambar 14 di atas merupakan respons server jika permintaan *login* berhasil.

3. Menambahkan Data AGV



Gambar 15. Permintaan Menambahkan Data AGV

Pada gambar 15 di atas, terlihat contoh permintaan (*request*) untuk menambahkan data AGV menggunakan metode POST ke *endpoint* `{{base_url}}/agvs`. Permintaan ini membutuhkan token akses yang didapatkan ketika *login* berhasil. Selain itu, permintaan ini juga membutuhkan *request body* yang berisi beberapa data yang diperlukan.

Gambar 16. Respons Berhasil Menambahkan Data AGV

Gambar 16 di atas merupakan respons server jika permintaan menambahkan data AGV berhasil.

4. Menambahkan Data Kode Pergerakan

Gambar 17. Permintaan Menambahkan Data Kode Pergerakan

Pada gambar 17 di atas, terlihat contoh permintaan (*request*) untuk menambahkan data kode pergerakan menggunakan metode POST ke *endpoint* `{{base_url}}/code-lists`. Permintaan ini membutuhkan token akses. Selain itu, permintaan ini juga membutuhkan *request body* yang berisi beberapa data yang diperlukan.

Gambar 18. Respons Berhasil Menambahkan Data Kode Pergerakan

Gambar 18 di atas merupakan respons server jika permintaan menambahkan data kode pergerakan berhasil.

5. Menambahkan Data RFID

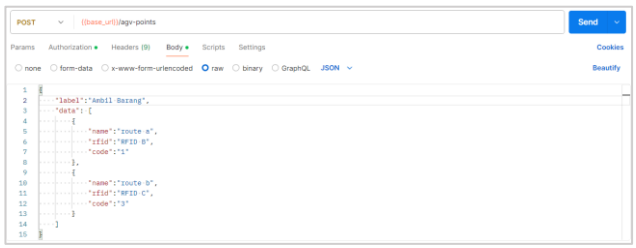
Gambar 19. Permintaan Menambahkan Data RFID

Pada gambar 19 di atas, terlihat contoh permintaan (*request*) untuk menambahkan data RFID menggunakan metode POST ke *endpoint* `{{base_url}}/rfids`. Permintaan ini membutuhkan token akses. Selain itu, permintaan ini juga membutuhkan *request body* yang berisi beberapa data yang diperlukan.

Gambar 20. Respons Berhasil Menambahkan Data RFID

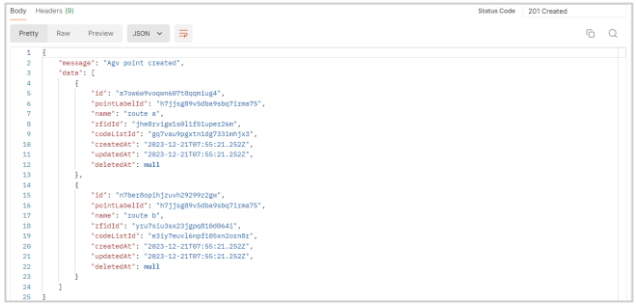
Gambar 20 di atas merupakan respons server jika permintaan menambahkan data RFID berhasil.

6. Menambahkan Data Rute



Gambar 21. Permintaan Menambahkan Data Rute

Pada gambar 21 di atas, terlihat contoh permintaan (*request*) untuk menambahkan data rute menggunakan metode POST ke *endpoint* `{{base_url}}/agv-points`. Permintaan ini membutuhkan token akses. Selain itu, permintaan ini juga membutuhkan *request body* yang berisi beberapa data yang diperlukan.



Gambar 22. Respons Berhasil Menambahkan Data Rute

Gambar 22 di atas merupakan respons server jika permintaan menambahkan data rute berhasil.

7. Menambahkan Data Rute AGV

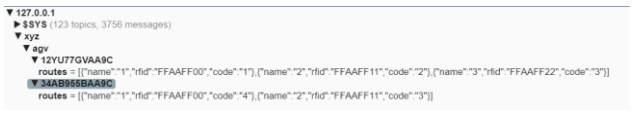
Gambar 23. Permintaan Menambahkan Data Rute AGV

Pada gambar 23 di atas, terlihat contoh permintaan (*request*) untuk menambahkan rute AGV menggunakan metode POST ke *endpoint* `{{base_url}}/agv-routes `. Permintaan ini membutuhkan token akses. Selain itu, permintaan ini juga membutuhkan *request body* yang berisi beberapa data yang diperlukan.

Gambar 24. Respons Berhasil Menambahkan Data Rute AGV

Gambar 24 di atas merupakan respons server jika permintaan menambahkan data rute AGV berhasil. Selain itu, seperti yang dilihat pada gambar 25, data rute akan dikirim ke *broker* MQTT dengan topik `xyz/agv/{{mac_address}}/routes`, di mana {{mac_address}} merupakan MAC *address* dari AGV terkait. Data rute yang berhasil dikirim ke MQTT *broker* dapat dilihat pada gambar 26 di bawah.

Gambar 25. MQTT *Broker* Sebelum Rute AGV Ditambahkan



Gambar 26. MQTT *Broker* Setelah Rute AGV Ditambahkan

3.4 Evaluasi

Evaluasi dilakukan dengan pengujian *black box* menggunakan 101 skenario pengujian yang dilaksanakan oleh pengembang REST API. Metode ini menekankan pada pengujian fungsionalitas sistem tanpa memeriksa struktur internal kodenya, sehingga memastikan bahwa permintaan yang dilakukan menghasilkan respons yang sesuai dengan spesifikasi yang telah ditentukan. Pengujian dilakukan pada REST API dan integrasi MQTT untuk memastikan keduanya berfungsi dengan baik dalam keseluruhan sistem. Dari pengujian yang telah dilakukan, diperoleh 101 keberhasilan dari 101 skenario. Dengan demikian, dapat disimpulkan bahwa pengujian *black box* pada REST API untuk manajemen rute AGV berhasil sepenuhnya dengan tingkat keberhasilan mencapai 100%. Semua skenario pengujian yang dirancang telah dijalankan dan hasilnya sesuai dengan harapan. Hasil pengujian ini menunjukkan

bahwa sistem yang dibangun mampu memenuhi spesifikasi dan kebutuhan pengguna secara efektif.

4. KESIMPULAN

Berdasarkan hasil penelitian yang berjudul "Rancang Bangun REST API untuk Manajemen Rute Robot AGV di PT XYZ", maka dapat disimpulkan sebagai berikut:

a. Perancangan dan pembangunan REST API untuk manajemen rute robot AGV berhasil dengan baik. REST API yang dikembangkan mampu mengelola data rute AGV, termasuk menambah, mengubah, dan menghapus rute dengan efisien. Seluruh fitur yang diperlukan untuk pengelolaan rute AGV telah diimplementasikan sesuai dengan spesifikasi yang ditetapkan, dan hasil pengujian menunjukkan tingkat keberhasilan 100% dari 101 skenario pengujian yang dijalankan.
b. Integrasi REST API dengan sistem AGV telah berhasil dilakukan dengan menggunakan MQTT *broker* untuk komunikasi antara API dan AGV. REST API mampu mempublikasikan data rute ke MQTT *broker* dengan topik yang sesuai, sehingga AGV dapat menerima dan menavigasi rute yang diberikan. Pengujian integrasi menunjukkan bahwa data rute yang dipublikasikan ke MQTT *broker* dapat diperbarui dan diterima dengan benar. Hal ini membuktikan bahwa integrasi REST API dengan sistem AGV berjalan sesuai dengan harapan dan dapat diandalkan untuk operasi AGV yang efisien.

Pada pengembangan REST API untuk manajemen rute robot AGV, masih terdapat beberapa kekurangan yang perlu diperbaiki. Berikut ini adalah beberapa saran terkait pengembangan REST API manajemen rute robot AGV untuk penelitian mendatang:

a. Menambahkan mekanisme kedaluwarsa pada token akses guna meningkatkan keamanan sistem. Token yang diberikan kepada pengguna harus memiliki masa berlaku tertentu, sehingga jika token tersebut dicuri atau disalahgunakan, dampaknya dapat diminimalkan.
b. Tingkatkan keamanan sistem dengan meningkatkan validasi dan penyaringan pada *input* pengguna untuk mencegah serangan injeksi dan memastikan data yang diterima valid dan aman.

DAFTAR PUSTAKA

[1] I. P. Himawati, H. Nopianti, dan D. Widiyati, “Sosialisasi Pengetahuan Mengenai Peluang dan Tantangan di Era Revolusi Industri 4.0 pada Pelajar di Sekolah Menengah Atas dan Kejuruan di Kota Bengkulu,” *Jurnal Widya Laksana*, vol. 9, no. 2, hlm. 205–212, 2020.

[2] M. Munadi, I. Haryanto, dan M. I. D. Surya, “Rancang Bangun dan Manufaktur Chassis Robot Automated Guided Vehicle (AGV) Sebagai Prototipe Alat Transportasi Barang pada Perusahaan Garmen,” *Jurnal Teknik Mesin S-1*, vol. 10, no. 2, hlm. 197–206, 2022.

[3] M. A. Latif, A. Rusdinar, dan R. Nugraha, “Perancangan dan Implementasi Automatic Guided Vehicle (AGV) Menggunakan Sistem Line Follower dan RFID Sebagai Pemetaan dengan Fuzzy Logic,” *e-Proceeding of Engineering*, vol. 6, no. 1, hlm. 95–102, 2019.

[4] W. G. Wardhana, I. Arwani, dan B. Rahayudi, “Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus: Fakultas Teknologi Pertanian Universitas Brawijaya),” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 2, hlm. 680–689, 2020, [Daring]. Tersedia pada: http://j-ptiik.ub.ac.id

[5] M. F. Nugroho, A. Primajaya, dan M. Jajuli, “Rancang Bangun REST API Aplikasi Manajemen Toko Menggunakan Nodejs pada Cantika Paint,” *Jurnal Mahasiswa Teknik Informatika*, vol. 7, no. 6, hlm. 3904–3910, 2023.

[6] Moh. S. Fikri, “Rancang Bangun Backend Sistem Manajemen Kehadiran Karyawan Menggunakan Framework Node Js Express (Studi Kasus: PT. Swa Nusa Multimedia),” Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Depok, 2023.

[7] Microsoft, “Apa itu PostgreSQL?” Diakses: 12 Maret 2024. [Daring]. Tersedia pada: https://azure.microsoft.com/id-id/resources/cloud-computing-dictionary/what-is-postgresql

[8] A. Izhar, “Rancang Bangun REST-API Menggunakan Express Js Guna Mencari Mentor Pribadi untuk Pembelajaran,” Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Depok, 2023.

[9] A. Kurnianto, J. D. Irawan, dan F. Ariwibisono, “Penerapan IoT (Internet of Things) untuk Controlling Lampu Menggunakan Protokol MQTT Berbasis Web,” *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 2, hlm. 1153–1161, 2022, [Daring]. Tersedia pada: https://www.embedded.com/

[10] T. Nordquist, “MQTT Explorer.” Diakses: 26 Maret 2024. [Daring]. Tersedia pada: https://mqtt-explorer.com/

[11] R. Aditya, V. H. Pranatawijaya, dan P. B. A. A. Putra, “Rancang Bangun Aplikasi Monitoring Kegiatan Menggunakan Metode Prototype,” *JOINTECOMS (Journal of Information Technology*

and Computer Science), vol. 1, no. 1, hlm. 47–57, 2021.

[12] F. Fahrullah, "Implementasi Pengujian Black Box pada Sistem Informasi Monitoring Akademik Dengan Pendekatan Teknik Equivalence Partitions.," *Jurnal Teknosains Kodepena*, vol. 01, no. 02, hlm. 94–100, 2021.

[13] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," *Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK*, 2020, [Daring]. Tersedia pada: https://www.researchgate.net/publication/346397070